

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Synthesizer Multi-Bus Component**

Inventor(s):

**Todor J. Fay**

**Brian Schmidt**

**Jim Geist**

ATTORNEY'S DOCKET NO. MS1-737US

1      **RELATED APPLICATIONS**

2      This application is related to a concurrently-filed U.S. Patent Application  
3      entitled "Audio Generation System Manager", to Todor Fay and Brian Schmidt,  
4      which is identified as client docket number MS1-723US, the disclosure of which  
5      is incorporated by reference herein.

6      This application is also related to a concurrently-filed U.S. Patent  
7      Application entitled "Accessing Audio Processing Components in an Audio  
8      Generation System", to Todor Fay and Brian Schmidt, which is identified as client  
9      docket number MS1-738US, the disclosure of which is incorporated by reference  
10     herein.

11     This application is also related to a concurrently-filed U.S. Patent  
12     Application entitled "Dynamic Channel Allocation in a Synthesizer Component",  
13     to Todor Fay, which is identified as client docket number MS1-739US, the  
14     disclosure of which is incorporated by reference herein.

15      **TECHNICAL FIELD**

16      This invention relates to audio processing and, in particular, to interfacing a  
17     synthesizer component with audio buffer components.

18      **BACKGROUND**

19      Multimedia programs present data to a user through both audio and video  
20     events while a user interacts with a program via a keyboard, joystick, or other  
21     interactive input device. A user associates elements and occurrences of a video  
22     presentation with the associated audio representation. A common implementation  
23     is to associate audio with movement of characters or objects in a video game.

1 When a new character or object appears, the audio associated with that entity is  
2 incorporated into the overall presentation for a more dynamic representation of the  
3 video presentation.

4 Audio representation is an essential component of electronic and  
5 multimedia products such as computer based and stand-alone video games,  
6 computer-based slide show presentations, computer animation, and other similar  
7 products and applications. As a result, audio generating devices and components  
8 are integrated with electronic and multimedia products for composing and  
9 providing graphically associated audio representations. These audio  
10 representations can be dynamically generated and varied in response to various  
11 input parameters, real-time events, and conditions. Thus, a user can experience  
12 the sensation of live audio or musical accompaniment with a multimedia  
13 experience.

14 Conventionally, computer audio is produced in one of two fundamentally  
15 different ways. One way is to reproduce an audio waveform from a digital sample  
16 of an audio source which is typically stored in a wave file (i.e., a .wav file). A  
17 digital sample can reproduce any sound, and the output is very similar on all sound  
18 cards, or similar computer audio rendering devices. However, a file of digital  
19 samples consumes a substantial amount of memory and resources for streaming  
20 the audio content. As a result, the variety of audio samples that can be provided  
21 using this approach is limited. Another disadvantage of this approach is that the  
22 stored digital samples cannot be easily varied.

23 Another way to produce computer audio is to synthesize musical instrument  
24 sounds, typically in response to instructions in a Musical Instrument Digital  
25 Interface (MIDI) file. MIDI is a protocol for recording and playing back music

1 and audio on digital synthesizers incorporated with computer sound cards. Rather  
2 than representing musical sound directly, MIDI transmits information and  
3 instructions about how music is produced. The MIDI command set includes note-  
4 on, note-off, key velocity, pitch bend, and other methods of controlling a  
5 synthesizer.

6 The audio sound waves produced with a synthesizer are those already  
7 stored in a wavetable in the receiving instrument or sound card. A wavetable is a  
8 table of stored sound waves that are digitized samples of actual recorded sound. A  
9 wavetable can be stored in read-only memory (ROM) on a sound card chip, or  
10 provided with software. Prestoring sound waveforms in a lookup table improves  
11 rendered audio quality and throughput. An advantage of MIDI files is that they  
12 are compact and require few audio streaming resources, but the output is limited to  
13 the number of instruments available in the designated General MIDI set and in the  
14 synthesizer, and may sound very different on different computer systems.

15 MIDI instructions sent from one device to another indicate actions to be  
16 taken by the controlled device, such as identifying a musical instrument (e.g.,  
17 piano, flute, drums, etc.) for music generation, turning on a note, and/or altering a  
18 parameter in order to generate or control a sound. In this way, MIDI instructions  
19 control the generation of sound by remote instruments without the MIDI control  
20 instructions carrying sound or digitized information. A MIDI sequencer stores,  
21 edits, and coordinates the MIDI information and instructions. A synthesizer  
22 connected to a sequencer generates audio based on the MIDI information and  
23 instructions received from the sequencer. Many sounds and sound effects are a  
24 combination of multiple simple sounds generated in response to the MIDI  
25 instructions.

1 A MIDI system allows audio and music to be represented with only a few  
2 digital samples rather than converting an analog signal to many digital samples.  
3 The MIDI standard supports different channels that can each simultaneously  
4 provide an output of audio sound wave data. There are sixteen defined MIDI  
5 channels, meaning that no more than sixteen instruments can be playing at one  
6 time. Typically, the command input for each channel represents the notes  
7 corresponding to an instrument. However, MIDI instructions can program a  
8 channel to be a particular instrument. Once programmed, the note instructions for  
9 a channel will be played or recorded as the instrument for which the channel has  
10 been programmed. During a particular piece of music, a channel can be  
11 dynamically reprogrammed to be a different instrument.

12 A Downloadable Sounds (DLS) standard published by the MIDI  
13 Manufacturers Association allows wavetable synthesis to be based on digital  
14 samples of audio content provided at run time rather than stored in memory. The  
15 data describing an instrument can be downloaded to a synthesizer and then played  
16 like any other MIDI instrument. Because DLS data can be distributed as part of an  
17 application, developers can be sure that the audio content will be delivered  
18 uniformly on all computer systems. Moreover, developers are not limited in their  
19 choice of instruments.

20 A DLS instrument is created from one or more digital samples, typically  
21 representing single pitches, which are then modified by a synthesizer to create  
22 other pitches. Multiple samples are used to make an instrument sound realistic  
23 over a wide range of pitches. DLS instruments respond to MIDI instructions and  
24 commands just like other MIDI instruments. However, a DLS instrument does not  
25 have to belong to the General MIDI set or represent a musical instrument at all.

1 Any sound, such as a fragment of speech or a fully composed measure of music,  
2 can be associated with a DLS instrument.

3 **Conventional Audio and Music System**

4 Fig. 1 illustrates a conventional audio and music generation system 100.  
5 The audio system 100 includes two discrete components, DirectMusic® 102 and  
6 DirectSound® 104. DirectMusic® and DirectSound® are application  
7 programming interfaces (APIs) available from Microsoft Corporation, Redmond  
8 Washington. DirectSound® plays prerecorded digital samples, typically from  
9 wave files, and DirectMusic® plays synthesized audio in response to MIDI files or  
10 preauthored musical segments.

11 The audio system 100 includes a synthesizer 106 having a synthesizer  
12 channel 108. Typically, a synthesizer is implemented in computer software, in  
13 hardware as part of a computer's internal sound card, or as an external device such  
14 as a MIDI keyboard or module. The synthesizer channel 108 is an audio data or  
15 communications path that represents a destination for a MIDI instruction. The  
16 channel 108 has a left and right audio data output, and a reverb audio data output.  
17 The reverb output is input to a reverb component 110, and the left and right audio  
18 data outputs are input to a left or right input component 112 and 114, respectively.  
19 The output of the reverb 110 is a stereo pair that is also input to the left or right  
20 input component 112 and 114, respectively. The synthesizer output 116 is a stereo  
21 pair that is input to a mixing component 118.

22 A MIDI instruction, such as a "note-on", directs a synthesizer 106 to play a  
23 particular note, or notes, on a synthesizer channel 108 having a designated  
24 instrument. The General MIDI standard defines standard sounds that can be  
25 combined and mapped into the sixteen separate instrument and sound channels. A

1 MIDI event on a synthesizer channel corresponds to a particular sound and can  
2 represent a keyboard key stroke, for example. The “note-on” MIDI instruction can  
3 be generated with a keyboard when a key is pressed and the “note-on” instruction  
4 is sent to synthesizer 106. When the key on the keyboard is released, a  
5 corresponding “note-off” instruction is sent to stop the generation of the sound  
6 corresponding to the keyboard key.

7 The audio system 100 includes a buffer component 120 that has multiple  
8 buffers 122(1...n). The output of the mixing component 118 associated with  
9 synthesizer channels 108 is input to one buffer 122(2) in the buffer component  
10 120. A buffer in this instance is typically an allocated area of memory that  
11 temporarily holds sequential samples of audio data that will be subsequently  
12 delivered to an audio rendering device such as a speaker.

13 An application program typically communicates with synthesizer 106 via  
14 some type of dedicated communication interface, commonly referred to as an API.  
15 In the audio system 100, an application program delivers audio content or other  
16 music events to the synthesizer 106. The audio content and music events are  
17 represented as data structures containing information about the audio content and  
18 music events such as pitch, relative volume, duration, and the like. Audio events  
19 are message-based data, such as MIDI files or musical segments, authored with an  
20 external device.

21 Sound effects can be implemented with a synthesizer, but the output is  
22 constrained to the stereo pair 116. For music generation, only having the ability to  
23 process audio in a synthesizer can be sufficient. In an audio system that supports  
24 both music and sound effects, however, a single output pair input to one buffer is a  
25 limitation to creating and enhancing the sound effects.

1

2 **SUMMARY**

3 An audio generation system produces streams of audio wave data and  
4 routes the audio wave data to audio buffers. The audio wave data is routed to the  
5 audio buffers via logic buses that correspond respectively to the audio buffers. A  
6 synthesizer, multiple synthesizers, and/or other streaming audio data sources  
7 produce the streams of audio wave data.

8 An audio buffer has one or more corresponding logic buses that route the  
9 audio wave data to the buffer. Each logic bus is assigned, or designated, to receive  
10 audio wave data from a source. When the source produces streams of audio wave  
11 data, the data is input to the assigned or designated logic buses.

12 A logic bus receives the audio wave data and routes it to the audio buffer  
13 corresponding to the logic bus. A logic bus can receive streams of audio wave  
14 data from multiple sources, and route the multiple audio wave data streams to an  
15 audio buffer. Additionally, an audio buffer can receive streams of audio wave  
16 data from multiple logic buses.

17

18 **BRIEF DESCRIPTION OF THE DRAWINGS**

19 The same numbers are used throughout the drawings to reference like  
20 features and components.

21 Fig. 1 is a block diagram that illustrates a conventional music generation  
22 system.

23 Fig. 2 is a block diagram that illustrates components of an audio generation  
24 system.

1 Fig. 3 is a block diagram that further illustrates components of the audio  
2 generation system shown in Fig. 2.

3 Fig. 4 is a block diagram of a data structure that correlates the components  
4 illustrated in Fig. 3.

5 Fig. 5 is a flow diagram of a method for an audio generation system with a  
6 multi-bus component.

7 Fig. 6 is a diagram of computing systems, devices, and components in an  
8 environment that can be used to implement the invention described herein.

9

10 **DETAILED DESCRIPTION**

11 The following description describes systems and methods to manage and  
12 route streams of audio wave data in an audio processing system. Audio wave data  
13 can be stored as a resource or generated by a synthesizer. Buffers receive and  
14 store the streams of audio wave data until it is recalled and processed or delivered  
15 to an audio rendering device such as a speaker. A multi-bus component is  
16 instantiated to route the streams of audio wave data generated by a synthesizer, or  
17 synthesizers, to the buffers. The configuration of the multi-bus component allows  
18 a stream of audio data output from a synthesizer to be routed to any number of  
19 buffers.

20 **Exemplary Audio Generation System**

21 Fig. 2 illustrates an audio generation system 200 having components that  
22 can be implemented within a computing device, or the components can be  
23 distributed within a computing system having more than one computing device.  
24 See the description of “Exemplary Computing System and Environment” below  
25 for specific examples and implementations of network and computing systems,

1 computing devices, and components that can be used to implement the invention  
2 described herein.

3 Audio generation system 200 includes an application program 202, audio  
4 sources 204, and an audio processing system 206. Application program 202 is one  
5 of a variety of different types of applications, such as a video game program, some  
6 other type of entertainment program, or an application that incorporates an audio  
7 representation with a video presentation.

8 Audio sources 204 supply digital samples of audio data such as from a  
9 wave file (i.e., a .wav file), message-based data such as from a MIDI file or a pre-  
10 authored segment file, or an audio sample such as a Downloadable Sound (DLS).  
11 Although not shown, the audio sources 204 can be stored in the application  
12 program 202 as a resource rather than in a separate file.

13 Application program 202 initiates that an audio source 204 be loaded and  
14 processed by the audio processing system 206. The application program 202  
15 interfaces with the other components of the audio generation system 200 via  
16 application programming interfaces (APIs). The various components described  
17 herein are implemented using standard programming techniques, including the use  
18 of OLE (object linking and embedding) and COM (component object model)  
19 interfaces. COM objects are implemented in a system memory of a computing  
20 device, each object having one or more interfaces, and each interface having one  
21 or more methods. The interfaces and interface methods can be called by  
22 application programs and by other objects. The interface methods of the objects  
23 are executed by a processing unit of the computing device. Familiarity with  
24 object-based programming, and with COM objects in particular, is assumed  
25 throughout this disclosure. However, those skilled in the art will recognize that

1 the audio generation systems and the various components described herein are not  
2 limited to a COM and/or OLE implementation, or to any other specific  
3 programming technique.

4 The audio processing system 206 converts an audio source 204 to a MIDI  
5 message format. Additional information regarding the audio data processing  
6 components described herein can be found in the concurrently-filed U.S. Patent  
7 Application entitled “Audio Generation System Manager”, which is incorporated  
8 by reference above. However, any audio processing system can be used to  
9 produce audio instructions for input to the audio generation system components.

10 The audio generation system 200 includes a synthesizer component 208, a  
11 multi-bus component 210, and audio buffers 212. Synthesizer component 208  
12 receives formatted MIDI messages from the audio processing system 206 and  
13 generates sound waveforms that can be played by a sound card, for example. The  
14 audio buffers 212 receive the audio wave data (i.e., sound waveforms) generated  
15 by the synthesizer 208 and streams the audio wave data in real-time to an audio  
16 rendering device. An audio buffer 212 can be designated in hardware or in  
17 software.

18 The multi-bus component 210 routes the audio wave data from the  
19 synthesizer component 208 to the audio buffers 212. The multi-bus component  
20 210 is implemented to represent actual studio audio mixing. In a studio, various  
21 audio sources such as instruments, vocals, and the like (which can also be outputs  
22 of a synthesizer) are input to a multi-channel mixing board that then routes the  
23 audio through various effects (e.g., audio processors), and then mixes the audio  
24 into the two channels that are a stereo signal. Additional information regarding  
25 the audio data processing components described herein can be found in the

1 concurrently-filed U.S. Patent Application entitled "Dynamic Channel Allocation  
2 in a Synthesizer Component", which is incorporated by reference above.

3 **Exemplary Synthesizer Multi-Bus Component**

4 Fig. 3 illustrates various components of the audio generation system 200 in  
5 accordance with an implementation of the invention described herein. Synthesizer  
6 component 208 has synthesizer channels 300(1-12). A synthesizer channel 300 is  
7 a communications path in the synthesizer 208 represented by a channel object. A  
8 channel object has APIs to receive and process MIDI events to generate audio  
9 wave data that is output by the synthesizer channels.

10 The synthesizer channels 300 are grouped into channel sets 302(1-3)  
11 according to the destination of the audio wave data that is output from each  
12 channel 300. Each channel set 302 has a channel designator 304 to identify the  
13 channels 300 corresponding to a particular channel set 302 within the synthesizer  
14 208. Channel set 302(1) includes synthesizer channels 300(1-3), channel set  
15 302(2) includes synthesizer channels 300(4-10), and channel set 302(3) includes  
16 synthesizer channels 300(11-12).

17 Multi-bus component 210 has multiple logical buses 306(1-4). A logical  
18 bus 306 is a logic connection or data communication path for audio wave data.  
19 Each logical bus 306 has a corresponding bus identifier (busID) 308 that uniquely  
20 identifies a particular logical bus. The logical buses 306 are configured to receive  
21 audio wave data from the synthesizer channels 300 and route the audio wave data  
22 to audio buffers component 212.

23 The audio buffers component 212 includes three buffers 310(1-3) that are  
24 consumers of the audio wave data. The audio buffers 310 receive audio wave data  
25 output from the logical buses 306 in the multi-bus component 210. An audio

1 buffer 310 receives an input of audio wave data from one or more logical buses  
2 306, and streams the audio wave data in real-time to a sound card or similar audio  
3 rendering device. Alternatively, an audio buffer 310 can process the audio wave  
4 data input with various effects-processing components (i.e., audio processing  
5 components) that corresponds to a designated function of a particular audio buffer  
6 310 before sending the audio data to be further processed and/or output as audible  
7 sound.

8 The audio buffers component 212 includes a two channel stereo buffer  
9 310(1) that receives audio wave data input from logic buses 306(1) and 306(2), a  
10 single channel mono buffer 310(2) that receives audio wave data input from logic  
11 bus 306(3), and a single channel reverb stereo buffer 310(3) that receives audio  
12 wave data input from logic bus 306(4).

13 Each logical bus 306 has a corresponding bus function identifier (funcID)  
14 312 that indicates the designated effects-processing function of the particular  
15 buffer 310 that receives the audio wave data output from the logical bus. For  
16 example, a bus funcID can indicate that the audio wave data output of a  
17 corresponding logical bus will be to a buffer 310 that functions as a left audio  
18 channel such as bus 306(1), a right audio channel such as bus 306(2), a mono  
19 channel such as bus 306(3), or a reverb channel such as bus 306(4). Additionally,  
20 a logical bus can output audio wave data to a buffer 310 that functions as a three-  
21 dimensional (3-D) audio channel, or output audio wave data to other types of  
22 effects-processing buffers.

23 Each channel set 302 in synthesizer 208 has a bus designator 314 to  
24 identify the logical buses 306 corresponding to a particular channel set 302. For  
25 example, synthesizer channels 300(1-3) of channel set 302(1) output audio wave

1 data that is designated as input for audio buffer 310(1). Audio buffer 310(1) is a  
2 two channel stereo buffer, thus having two associated buses 306(1) and 306(2) that  
3 input audio wave data to the buffer. The channel set 302(1) bus designator 314  
4 designates buses 1 and 2 as the destination for audio wave data output from  
5 channels 300(1-3) in the channel set.

6 Channel set 302(2) includes synthesizer channels 300(4-10) that output  
7 audio wave data designated as input for audio buffer 310(2). Audio buffer 310(2)  
8 is a single channel mono buffer and has one associated bus 306(3) that inputs  
9 audio wave data to the buffer. The channel set 302(2) bus designator 314  
10 designates bus 3 as the destination for audio wave data output from synthesizer  
11 channels 300(4-10). Channel set 302(3) includes synthesizer channels 300(11-12)  
12 that output audio wave data designated as input for audio buffers 310(1) and  
13 310(3). The channel set 302(3) bus designator 314 designates buses 1, 2, and 4 as  
14 the destination for audio wave data output from synthesizer channels 300(11-12).

15 A logical bus 306 can have more than one input, from more than one  
16 synthesizer, synthesizer channel, and/or audio source. A synthesizer 208 can mix  
17 audio wave data by routing one output from a synthesizer channel 300 to any  
18 number of logical buses 306 in the multi-bus component 210. For example,  
19 logical bus 306(1) has multiple inputs from synthesizer channels 300(1-3) and  
20 300(11-12). Each logical bus 306 outputs audio wave data to one associated  
21 buffer 310, but a particular buffer 310 can have more than one input from different  
22 logical buses. For example, logical buses 306(1) and 306(2) output audio wave  
23 data to one designated buffer. The designated audio buffer 310(1), however,  
24 receives the audio wave data output from both buses.

1        Although the multi-bus component 210 is shown having only four logical  
2        buses 306(1-4), it is to be appreciated that the logical buses are dynamically  
3        created as needed, and released when no longer needed. Thus, the multi-bus  
4        component 210 can support any number of logical buses at any one time as needed  
5        to route audio wave data from the synthesizer 208 to the audio buffers 310.  
6        Similarly, it is to be appreciated that there can be any number of audio buffers 310  
7        available to receive audio wave data at any one time. Furthermore, although the  
8        multi-bus component 210 is shown as an independent component, it can be  
9        integrated with the synthesizer component 208, or the audio buffers component  
10       212.

11       Fig. 4 illustrates a bus active list 400 that is a data structure maintained by  
12       the computing device that implements the multi-bus component 210. The active  
13       list 400 is a bus-to-buffer mapping list having a plurality of mappings. Each  
14       mapping has a bus identifier (busID) 402, a function identifier (funcID) 404, and a  
15       pointer 406. Thus, each mapping associates a busID with a bus funcID.  
16       Additionally, active list 400 associates a pointer 406 to the buffer 310 that  
17       corresponds to a particular busID 402. Those skilled in the art will recognize that  
18       various techniques are available to implement the bus active list 400 as a data  
19       structure.

20       Audio wave data is routed from the synthesizer channels 300 to the audio  
21       buffers 310 based on a “pull model”. That is, when an audio buffer 310 is  
22       available to receive audio wave data, the buffer requests output from the  
23       synthesizer 208. The synthesizer 208 receives the request for audio wave data  
24       from an audio buffer 310 along with a busID for the bus, or busIDs for the buses,  
25       that correspond to the buffer. The active list 400 is passed to the synthesizer 208

1 when a buffer 310 requests the audio wave data. The synthesizer 208 determines  
2 the associated funcID 404 of each logical bus corresponding to the available buffer  
3 and then designates which synthesizer channels 300 will output the audio wave  
4 data to the corresponding bus, or buses.

5 **File Format and Component Instantiation**

6 Configuration information for the synthesizer component 208, the multi-  
7 bus component 210, and the audio buffers component 212 is stored in a well-  
8 known format such as the Resource Interchange File Format (RIFF). A RIFF file  
9 includes a file header followed by what are known as “chunks.” The file header  
10 contains data describing an audio buffer object, for example, such as a buffer  
11 identifier, descriptor, the buffer function and associated effects (i.e., audio  
12 processors), and corresponding busIDs.

13 Each of the chunks following a file header corresponds to a data item that  
14 describes the object, such as an audio buffer object effect. Each chunk consists of  
15 a chunk header followed by actual chunk data. A chunk header specifies an object  
16 class identifier (CLSID) that can be used for creating an instance of the object.  
17 Chunk data consists of the audio buffer effect data to define the audio buffer  
18 configuration.

19 Audio buffers are created in accordance with configurations defined by  
20 RIFF files. A RIFF file for a buffer configuration includes a buffer global unique  
21 identifier (buffer GUID), a buffer descriptor, and bus identifier (busID) data. The  
22 buffer GUID uniquely identifies each buffer. A buffer GUID can be used to  
23 determine which synthesizer channels connect to which buffers. By using a  
24 unique buffer GUID for each buffer, different synthesizer channels, and channels

1 from different synthesizers, can connect to the same buffer or uniquely different  
2 ones, whichever is preferred.

3 The buffer descriptor defines how many audio channels a buffer will have  
4 as well as initial settings for volume, and the like. The busID data designates a  
5 logic bus, or buses, that connect to a particular buffer. There can be any number  
6 of logic buses connected to a particular buffer to input audio wave data. For  
7 example, stereo buffer 310(1) (Fig. 3) is a two channel stereo buffer and has two  
8 busIDs: a busID\_LEFT corresponding to logic bus 306(1) and a busID\_RIGHT  
9 corresponding to logic bus 306(2).

10 A RIFF file for a synthesizer configuration defines the synthesizer channels  
11 and includes both a synthesizer channel-to-buffer assignment list and a buffer  
12 configuration list stored in the synthesizer configuration data. The synthesizer  
13 channel-to-buffer assignment list defines the synthesizer channel sets and the  
14 buffers that are designated as the destination for audio wave data output from the  
15 synthesizer channels in the channel set. The assignment list associates buffers  
16 according to buffer GUIDs which are defined in the buffer configuration list.

17 Defining the buffers by buffer GUIDs facilitates the synthesizer channel-to-  
18 buffer assignments to identify which buffer will receive audio wave data from a  
19 synthesizer channel. Defining buffers by buffer GUIDs also facilitates sharing  
20 resources. More than one synthesizer can output audio wave data to the same  
21 audio buffer. When an audio buffer is instantiated, or provided, for use by a first  
22 synthesizer, a second synthesizer can output audio wave data to the buffer if it is  
23 available to receive data input. The buffer configuration list also maintains flag  
24 indicators that indicate whether a particular buffer can be a shared resource or not.

1        The buffer and synthesizer configurations support COM interfaces for  
2 reading and loading the data from a file. To instantiate, or provide, a synthesizer  
3 component 208 and/or an audio buffer 310, an application program 202 first  
4 instantiates a component using a COM function. The application program then  
5 calls a load method for a synthesizer object or a buffer object, and specifies a RIFF  
6 file stream. The object parses the RIFF file stream and extracts header  
7 information. When it reads individual chunks, it creates corresponding synthesizer  
8 channel objects or a buffer object based on the chunk header information.  
9 However, those skilled in the art will recognize that the audio generation systems  
10 and the various components described herein are not limited to a COM  
11 implementation, or to any other specific programming technique.

12        **Method for an Exemplary Audio Generation System**

13        Fig. 5 illustrates a method for an audio generation system with a multi-bus  
14 component and refers to components described in Figs. 2-4 by reference number.  
15 The order in which the method is described is not intended to be construed as a  
16 limitation. Furthermore, the method can be implemented in any suitable hardware,  
17 software, firmware, or combination thereof.

18        At block 500, a synthesizer component is provided. For example, the  
19 synthesizer component can be instantiated from a synthesizer configuration file  
20 format (e.g., a RIFF file as described above). A synthesizer component can also  
21 be created from a file representation that is loaded and stored in a synthesizer  
22 configuration object that maintains all of the information defined in the  
23 synthesizer configuration file format. Alternatively, a synthesizer component can  
24 be created directly by an audio rendition manager.

1 At block 502, audio buffers are provided. For example, the audio buffers  
2 can be instantiated from a buffer configuration file format (e.g., a RIFF file).  
3 Alternatively, an audio buffer component can be created from a file representation  
4 that is loaded and stored in a buffer configuration object that maintains all of the  
5 information defined in the buffer configuration file format. The information  
6 includes the number of buffer channels, a buffer GUID, and the busID data that  
7 indicates the funcID of each logical bus that connects to the audio buffer.

8 At block 504, a multi-bus component is provided having logical buses  
9 corresponding to the audio buffers created at block 502. For example, stereo  
10 buffer 310(1) is created and logical buses 306(1) and 306(2) corresponding to  
11 stereo buffer 310(1) are instantiated. At block 506, a bus-to-buffer mapping list,  
12 such as bus active list 400 for example, is created to reflect which logical buses  
13 correspond to an audio buffer.

14 At block 508, synthesizer channels are allocated in the synthesizer. The  
15 synthesizer channels can be allocated according to a synthesizer channel-to-buffer  
16 assignment list stored in the synthesizer configuration data. At block 510, the  
17 synthesizer (instantiated at block 500) receives a request from an audio buffer for  
18 audio wave data to process. The request for audio wave data includes the bus-to-  
19 buffer mapping list (created at block 506). At block 512, the synthesizer  
20 determines the function of the requesting audio buffer from the associated funcIDs  
21 for each corresponding logic bus listed in the bus-to-buffer mapping list.

22 At block 514, the synthesizer channels are assigned to buffers according to  
23 corresponding buffer GUIDs maintained in a buffer configuration list which is  
24 stored with the synthesizer configuration file data. At block 516, the synthesizer  
25 determines which logic buses correspond to each buffer that has been assigned to a

1 synthesizer channel (at block 514). At block 518, the synthesizer associates a bus  
2 designator for each synthesizer channel to indicate which bus or buses correspond  
3 to a particular synthesizer channel audio wave data output. For example, bus  
4 designator 314 indicates that synthesizer channels 300(1-3) of channel set 302(1)  
5 are associated with logical buses 1 and 2.

6 At block 520, the synthesizer determines which synthesizer channels can  
7 output audio wave data to the audio buffer that has a function type defined by the  
8 funcID corresponding to the associated logical bus or buses. At block 522, audio  
9 data is routed from the synthesizer 208, through logical buses 306 in the multi-bus  
10 component 210, and to audio buffers 310 in the audio buffers component 212.

### 11 Exemplary Computing System and Environment

12 Fig. 6 illustrates an example of a computing environment 600 within which  
13 the computer, network, and system architectures described herein can be either  
14 fully or partially implemented. Exemplary computing environment 600 is only  
15 one example of a computing system and is not intended to suggest any limitation  
16 as to the scope of use or functionality of the network architectures. Neither should  
17 the computing environment 600 be interpreted as having any dependency or  
18 requirement relating to any one or combination of components illustrated in the  
19 exemplary computing environment 600.

20 The computer and network architectures can be implemented with  
21 numerous other general purpose or special purpose computing system  
22 environments or configurations. Examples of well known computing systems,  
23 environments, and/or configurations that may be suitable for use include, but are  
24 not limited to, personal computers, server computers, thin clients, thick clients,  
25 hand-held or laptop devices, multiprocessor systems, microprocessor-based

1 systems, set top boxes, programmable consumer electronics, network PCs,  
2 minicomputers, mainframe computers, gaming consoles, distributed computing  
3 environments that include any of the above systems or devices, and the like.

4 Implementing a multi-bus component may be described in the general  
5 context of computer-executable instructions, such as program modules, being  
6 executed by a computer. Generally, program modules include routines, programs,  
7 objects, components, data structures, etc. that perform particular tasks or  
8 implement particular abstract data types. Implementing a multi-bus component  
9 may also be practiced in distributed computing environments where tasks are  
10 performed by remote processing devices that are linked through a communications  
11 network. In a distributed computing environment, program modules may be  
12 located in both local and remote computer storage media including memory  
13 storage devices.

14 The computing environment 600 includes a general-purpose computing  
15 system in the form of a computer 602. The components of computer 602 can  
16 include, by are not limited to, one or more processors or processing units 604, a  
17 system memory 606, and a system bus 608 that couples various system  
18 components including the processor 604 to the system memory 606.

19 The system bus 608 represents one or more of any of several types of bus  
20 structures, including a memory bus or memory controller, a peripheral bus, an  
21 accelerated graphics port, and a processor or local bus using any of a variety of  
22 bus architectures. By way of example, such architectures can include an Industry  
23 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an  
24 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)  
25

1 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a  
2 Mezzanine bus.

3 Computer system 602 typically includes a variety of computer readable  
4 media. Such media can be any available media that is accessible by computer 602  
5 and includes both volatile and non-volatile media, removable and non-removable  
6 media. The system memory 606 includes computer readable media in the form of  
7 volatile memory, such as random access memory (RAM) 610, and/or non-volatile  
8 memory, such as read only memory (ROM) 612. A basic input/output system  
9 (BIOS) 614, containing the basic routines that help to transfer information  
10 between elements within computer 602, such as during start-up, is stored in ROM  
11 612. RAM 610 typically contains data and/or program modules that are  
12 immediately accessible to and/or presently operated on by the processing unit 604.

13 Computer 602 can also include other removable/non-removable,  
14 volatile/non-volatile computer storage media. By way of example, Fig. 6  
15 illustrates a hard disk drive 616 for reading from and writing to a non-removable,  
16 non-volatile magnetic media (not shown), a magnetic disk drive 618 for reading  
17 from and writing to a removable, non-volatile magnetic disk 620 (e.g., a “floppy  
18 disk”), and an optical disk drive 622 for reading from and/or writing to a  
19 removable, non-volatile optical disk 624 such as a CD-ROM, DVD-ROM, or other  
20 optical media. The hard disk drive 616, magnetic disk drive 618, and optical disk  
21 drive 622 are each connected to the system bus 608 by one or more data media  
22 interfaces 626. Alternatively, the hard disk drive 616, magnetic disk drive 618,  
23 and optical disk drive 622 can be connected to the system bus 608 by a SCSI  
24 interface (not shown).

1        The disk drives and their associated computer-readable media provide non-  
2 volatile storage of computer readable instructions, data structures, program  
3 modules, and other data for computer 602. Although the example illustrates a  
4 hard disk 616, a removable magnetic disk 620, and a removable optical disk 624,  
5 it is to be appreciated that other types of computer readable media which can store  
6 data that is accessible by a computer, such as magnetic cassettes or other magnetic  
7 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or  
8 other optical storage, random access memories (RAM), read only memories  
9 (ROM), electrically erasable programmable read-only memory (EEPROM), and  
10 the like, can also be utilized to implement the exemplary computing system and  
11 environment.

12        Any number of program modules can be stored on the hard disk 616,  
13 magnetic disk 620, optical disk 624, ROM 612, and/or RAM 610, including by  
14 way of example, an operating system 626, one or more application programs 628,  
15 other program modules 630, and program data 632. Each of such operating  
16 system 626, one or more application programs 628, other program modules 630,  
17 and program data 632 (or some combination thereof) may include an embodiment  
18 of a implementing a multi-bus component.

19        Computer system 602 can include a variety of computer readable media  
20 identified as communication media. Communication media typically embodies  
21 computer readable instructions, data structures, program modules, or other data in  
22 a modulated data signal such as a carrier wave or other transport mechanism and  
23 includes any information delivery media. The term “modulated data signal”  
24 means a signal that has one or more of its characteristics set or changed in such a  
25 manner as to encode information in the signal. By way of example, and not

1 limitation, communication media includes wired media such as a wired network or  
2 direct-wired connection, and wireless media such as acoustic, RF, infrared, and  
3 other wireless media. Combinations of any of the above are also included within  
4 the scope of computer readable media.

5 A user can enter commands and information into computer system 602 via  
6 input devices such as a keyboard 634 and a pointing device 636 (e.g., a “mouse”).  
7 Other input devices 638 (not shown specifically) may include a microphone,  
8 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and  
9 other input devices are connected to the processing unit 604 via input/output  
10 interfaces 640 that are coupled to the system bus 608, but may be connected by  
11 other interface and bus structures, such as a parallel port, game port, or a universal  
12 serial bus (USB).

13 A monitor 642 or other type of display device can also be connected to the  
14 system bus 608 via an interface, such as a video adapter 644. In addition to the  
15 monitor 642, other output peripheral devices can include components such as  
16 speakers (not shown) and a printer 646 which can be connected to computer 602  
17 via the input/output interfaces 640.

18 Computer 602 can operate in a networked environment using logical  
19 connections to one or more remote computers, such as a remote computing device  
20 648. By way of example, the remote computing device 648 can be a personal  
21 computer, portable computer, a server, a router, a network computer, a peer device  
22 or other common network node, and the like. The remote computing device 648 is  
23 illustrated as a portable computer that can include many or all of the elements and  
24 features described herein relative to computer system 602.

1       Logical connections between computer 602 and the remote computer 648  
2 are depicted as a local area network (LAN) 650 and a general wide area network  
3 (WAN) 652. Such networking environments are commonplace in offices,  
4 enterprise-wide computer networks, intranets, and the Internet. When  
5 implemented in a LAN networking environment, the computer 602 is connected to  
6 a local network 650 via a network interface or adapter 654. When implemented in  
7 a WAN networking environment, the computer 602 typically includes a modem  
8 656 or other means for establishing communications over the wide network 652.  
9 The modem 656, which can be internal or external to computer 602, can be  
10 connected to the system bus 608 via the input/output interfaces 640 or other  
11 appropriate mechanisms. It is to be appreciated that the illustrated network  
12 connections are exemplary and that other means of establishing communication  
13 link(s) between the computers 602 and 648 can be employed.

14      In a networked environment, such as that illustrated with computing  
15 environment 600, program modules depicted relative to the computer 602, or  
16 portions thereof, may be stored in a remote memory storage device. By way of  
17 example, remote application programs 658 reside on a memory device of remote  
18 computer 648. For purposes of illustration, application programs and other  
19 executable program components, such as the operating system, are illustrated  
20 herein as discrete blocks, although it is recognized that such programs and  
21 components reside at various times in different storage components of the  
22 computer system 602, and are executed by the data processor(s) of the computer.

23      **Conclusion**

24      A synthesizer and multi-bus component allows an application program to  
25 specify, for example, unique 3-D positions for as many video entities as the

application requires for audio representation corresponding to a video presentation. Specifically, this is accomplished by routing audio wave data from a synthesizer channel to multiple buffers via logic buses that connect to the multiple buffers having mixing and/or 3-D effects-processing.

Interfacing the synthesizer and the audio buffers with the logic buses of the multi-bus component allows for greater flexibility in routing the audio wave data generated by the synthesizer. The flexibility in routing also means that groups of instruments can be routed through different buffers with different effects-processing, or sound effects can be sub-mixed and routed to unique 3-D positions.

Although the systems and methods have been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.